# Лабораторная работа № 8. Функции в языке С. Введение

## Задание 1

1. Даны три одномерных массива вещественных чисел A[1 . . . 6], A[1 . . . 8] и C[1 . . . 7]. Найти общую сумму положительных элементов в массивах. Нахождение суммы элементов в массиве оформить функцией.
2. –

|  |  |  |
| --- | --- | --- |
| Идентификатор | Семантика | Тип |
| n, k, m | Размеры массивов | int |
| min\_range, max\_range | Границы диапазона случайных чисел | int |
| filling | Функция запыления массива случайными числами | - |
| sum\_of\_positive | Функция нахождение суммы положительных элементов | double |
| a[n], b[k], c[m] | Массивы | double |
| sums | Сумма всех положительных элементов | double |
| i | Счётчик цикла | int |

#include <stdio.h>  
#include <stdlib.h>  
#include <time.h>  
  
#define n 6 // Размер первого массива  
#define k 8 // Размер второго массива  
#define m 7 // Размер третьего массива  
#define min\_range -1000 //Нижняя граница диапазона случайных чисел  
#define max\_range 1000 // Верхняя граница диапазона случайных чисел  
  
// Функция заполнения массива случайными числами  
void filling(double \*array, int amount\_of\_elements) {  
 srand(time(NULL));  
 for (int i = 0; i < amount\_of\_elements; i++) {  
 array[i] = (double)(rand() % (max\_range - min\_range + 1)) / 100 - 10;  
 }  
}  
  
// Функция нахождение суммы положительных элементов  
double sum\_of\_positive(double \*array, int amount\_of\_elements) {  
 double sum = 0;  
 int i;  
 for (i = 0; i < amount\_of\_elements; i++) {  
 if (array[i] >= 0)  
 sum += array[i];  
 }  
 return sum;  
}  
  
int main() {  
 double a[n], b[k], c[m];  
 int i;  
 // Заполнение массивов случайными вещественными числами  
 filling(a, n);  
 filling(b, k);  
 filling(c, m);  
 // Нахождение сумм всех положительных элементов  
 double sums = sum\_of\_positive(a, n) + sum\_of\_positive(b, k) + sum\_of\_positive(c, m);  
 printf("Сумма всех положительных элементов: %2.2f.", sums);  
 return 0;  
}

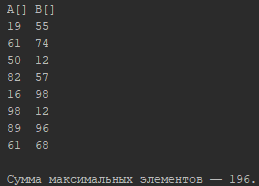
1. Сумма всех положительных элементов: 52.96.

## Задание 2

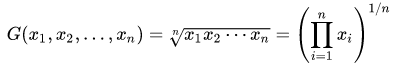
1. Даны два одномерных массива целых чисел A[1 . . . 8] и B[1 . . . 8]. Найти сумму их максимальных элементов. Для нахождения максимального элемента в массиве использовать функцию.
2. –

|  |  |  |
| --- | --- | --- |
| Идентификатор | Семантика | Тип |
| n | Размер массива | int |
| max\_range | Граница диапазона случайных чисел | int |
| filling | Функция заполнения массива случайными числами | - |
| sum\_of\_max | Функция нахождения суммы максимума двух массивов одного размера | int |
| a[n], b[n] | Массивы | int |

#include <stdio.h>  
#include <stdlib.h>  
#include <time.h>  
  
#define n 8 // Размер первого и второго массива  
#define max\_range 100 //Верхняя граница диапазона случайных чисел  
  
// Функция заполнения массива случайными числами  
void filling(int \*array, int amount\_of\_elements) {  
 for (int i = 0; i < amount\_of\_elements; i++) {  
 array[i] = rand() % max\_range;  
 }  
}  
// Функция нахождения суммы максимума двух массивов одного размера  
int sum\_of\_max(int \*array1, int \*array2, int amount\_of\_elements) {  
 int max1 = array1[0], max2 = array2[0];  
 for (int i = 0; i < amount\_of\_elements; i++) {  
 if (max1 < array1[i])  
 max1 = array1[i];  
 if (max2 < array2[i])  
 max2 = array2[i];  
 }  
 return max1 + max2;  
}  
  
int main(int argc, char const \*argv[]) {  
 srand(time(NULL));  
 int a[n], b[n];  
 // Заполнение массивов случайными числами  
 filling(a, n);  
 filling(b, n);  
 printf("A[]\tB[]\n");  
 for (int i = 0; i < n; i++) {  
 printf("%d\t%d\n", a[i], b[i]);  
 }  
 printf("\nСумма максимальных элементов — %d.", sum\_of\_max(a, b, n));  
 return 0;  
}

1. 

## Задание 3

1. Даны три одномерных массива вещественных чисел A[1 . . . 6], V [1 . . . 8] и C[1 . . . 7]. Найти среднее геометрическое значение положительных элементов для каждого. Вычисление среднего геометрического оформить в виде функции.
2. 

|  |  |  |
| --- | --- | --- |
| Идентификатор | Семантика | Тип |
| n, k ,m | Размеры массивов | int |
| min\_range, max\_range | Границы диапазона случайных чисел | int |
| filling | Функция заполнения массива случайными числами | - |
| geometric\_mean | Функция нахождения среднего геометрического | double |
| a[n], b[k], c[m] | Массивы | double |

#include <math.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <time.h>  
  
#define n 6 // Размер первого массива  
#define k 8 // Размер второго массива  
#define m 7 // Размер третьего массива  
#define min\_range -1000 // Нижняя граница диапазона случайных чисел  
#define max\_range 1000 // Верхняя граница диапазона случайных чисел  
  
// Функция заполнения массива случайными числами  
void filling(double \*array, int amount\_of\_elements) {  
 for (int i = 0; i < amount\_of\_elements; i++) {  
 array[i] = (double)(rand() % (max\_range - min\_range + 1)) / 100;  
 }  
}  
// Функция нахождения среднего геометрического  
double geometric\_mean(double \*array, int amount\_of\_elements) {  
 double product = 1, mean;  
 for (int i = 0; i < amount\_of\_elements; i++)  
 product \*= array[i];  
 mean = pow(product, 1.0 / amount\_of\_elements);  
 return mean;  
}  
  
int main(int argc, char const \*argv[]) {  
 srand(time(NULL));  
 double a[n], b[k], c[m];  
 int i;  
 // Заполнение массивов случайными вещественными числами  
 filling(a, n);  
 filling(b, k);  
 filling(c, m);  
 printf("Массивы:\nA[]\tB[]\tC[]\n");  
 printf("%2.2f\t%2.2f\t%2.2f", geometric\_mean(a, n), geometric\_mean(b, k), geometric\_mean(c, m));  
 return 0;  
}

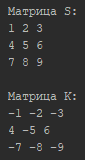
1. 

## Задание 4

1. Даны две матрицы целых чисел S[1 . . . 3, 0 . . . 2], K[1 . . . 3, 0 . . . 2], в каждой из которых имеется по два одинаковых числа. Распечатать их значения.
2. –

|  |  |  |
| --- | --- | --- |
| Идентификатор | Семантика | Тип |
| n | размер массива | int |
| printing | Функция, распечатывающая значения массивов | int |
| s[n][n], k[n][n] | Матрицы | int |

#include <stdio.h>  
#include <stdlib.h>  
#include <time.h>  
  
#define n 3 // Размеры первого и второго массива  
  
void printing(int array[][n]) {  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {  
 printf("%d ", array[i][j]);  
 }  
 printf("\n");  
 }  
}  
  
int main() {  
 int s[n][n]={1, 2, 3, 4, 5, 6, 7, 8, 9};  
 int k[n][n]={-1, -2, -3, 4, -5, 6, -7, -8, -9};  
 // Вывод матрицы s  
 printf("Матрица S:\n" );  
 printing(s);  
 // Вывод матрицы k  
 printf("\nМатрица K:\n" );  
 printing(k);  
 return 0;  
}

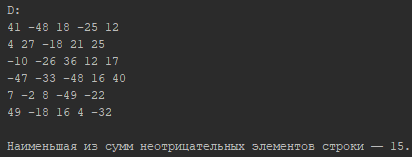
1. 

## Задание 5

1. Дана матрица целых чисел D[1 . . . 6, 1 . . . 5]. Найти наименьшую из сумм неотрицательных элементов строк матрицы. Для вычисления суммы использовать подпрограмму (функцию).
2. –

|  |  |  |
| --- | --- | --- |
| Идентификатор | Семантика | Тип |
| n, m | Размер массива | int |
| max\_range | Граница диапазона случайных чисел | int |
| filling | Функция заполнения массива случайными числами | - |
| sum\_of\_strings | Функция нахождения суммы положительных элементов в строке | int |
| d[n] | Массив | int |
| minimal\_sum | Минимальная сумма элементов строки | int |

#include <stdio.h>  
#include <stdlib.h>  
#include <time.h>  
  
#define n 6 // Количество строк  
#define m 5 // Количество столбцов  
#define max\_range 100 // Верхняя граница диапазона случайных чисел  
  
// Функция заполнения массива случайными числами  
void filling(int array[n][m]) {  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < m; j++) {  
 array[i][j] = rand() % max\_range - max\_range / 2;  
 }  
 }  
}  
  
// Функция вывода матрицы  
void printing(int array[n][m]) {  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < m; j++) {  
 printf("%d ", array[i][j]);  
 }  
 printf("\n");  
 }  
}  
  
// Функция нахождения суммы минимальных неотрицательных элементов  
int sum\_of\_strings(int array[n][m], int i) {  
 int sum = 0;  
 for (int j = 0; j < m; j++)  
 if (array[i][j] > 0)  
 sum += array[i][j];  
return sum;  
}  
  
int main() {  
 srand(time(NULL));  
 int d[n][m];  
 filling(d);  
 printf("D:\n");  
 printing(d);  
 int minimal\_sum = sum\_of\_strings(d, 0);  
 for (int i = 1; i < n; i++)  
 if (minimal\_sum > sum\_of\_strings(d, i))  
 minimal\_sum = sum\_of\_strings(d, i);  
 printf("\nНаименьшая из сумм неотрицательных элементов строки — %d.",  
 minimal\_sum);  
 return 0;  
}

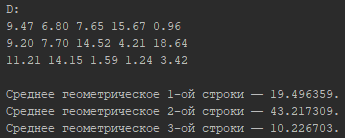
1. 

## Задание 6

1. Дана матрица целых чисел D[1 . . . 3, 1 . . . 5]. Используя функцию, найти среднее геометрическое значение для каждой строки матрицы.
2. –

|  |  |  |
| --- | --- | --- |
| Идентификатор | Семантика | Тип |
| n, m | Размер массива | int |
| max\_range, min\_range | Граница диапазона случайных чисел | int |
| filling | Функция заполнения массива случайными числами | - |
| geometric mean | Функция нахождения среднего геометрического для каждой строки | double |
| printing | Функция вывода матрицы | - |
| d[n][m] | Массив | double |

#include <stdio.h>  
#include <stdlib.h>  
#include <math.h>  
#include <time.h>  
  
#define n 3 // Количество строк  
#define m 5 // Количество столбцов  
#define min\_range -1000 // Нижняя граница диапазона случайных чисел  
#define max\_range 1000 // Верхняя граница диапазона случайных чисел  
  
// Функция заполнения массива случайными числами  
void filling(double array[n][m]) {  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < m; j++) {  
 array[i][j] = (double)(rand() % (max\_range - min\_range + 1)) / 100;  
 }  
 }  
}  
  
// Функция вывода матрицы  
void printing(double array[n][m]) {  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < m; j++) {  
 printf("%2.2f ", array[i][j]);  
 }  
 printf("\n");  
 }  
}  
  
// Функция нахождения среднего геометрического для каждой строки  
double geometric\_mean(double array[n][m], int i) {  
 double product = 1, mean;  
 for (int j = 0; j < m; j++)  
 product \*= array[i][j];  
 mean = pow(product, 1.0 / n);  
 return mean;  
}  
  
int main() {  
 srand(time(NULL));  
 double d[n][m];  
 filling(d);  
 printf("D:\n");  
 printing(d);  
 for (int i = 0; i < n; i++)  
 printf("\nСреднее геометрическое %d-ой строки — %f.", i + 1, geometric\_mean(d, i));  
 return 0;  
}

1. 

## Задание 7

1. Дана матрица целых чисел F[1 . . . 4, 1 . . . 5]. Найти наименьшие значения элементов в каждой из строк матрицы с помощью функции.
2. –

|  |  |  |
| --- | --- | --- |
| Идентификатор | Семантика | Тип |
| n, m | Размер массива | int |
| max\_range | Граница диапазона случайных чисел | int |
| filling | Функция заполнения массива случайными числами | - |
| min\_of\_string | Функция нахождения минимального элемента в строке | int |
| printing | Функция вывода матрицы | - |
| f[n][m] | Массив | int |

#include <stdio.h>  
#include <stdlib.h>  
#include <math.h>  
#include <time.h>  
  
#define n 4 // Количество строк  
#define m 5 // Количество столбцов  
#define max\_range 100 // Верхняя граница диапазона случайных чисел  
  
// Функция заполнения массива случайными числами  
void filling(int array[n][m]) {  
 for (int i = 0; i < n; i++)  
 for (int j = 0; j < m; j++)  
 array[i][j] = rand() % max\_range;  
}  
  
// Функция вывода матрицы  
void printing(int array[n][m]) {  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < m; j++) {  
 printf("%d ", array[i][j]);  
 }  
 printf("\n");  
 }  
}  
  
// Функция нахождения минимального элемента в строке  
int min\_of\_string(int array[n][m], int i) {  
 int min = array[i][0];  
 for (int j = 0; j < m; j++)  
 if (min > array[i][j])  
 min = array[i][j];  
 return min;  
}  
  
int main() {  
 srand(time(NULL));  
 int f[n][m];  
 filling(f);  
 printf("F:\n");  
 printing(f);  
 for (int i = 0; i < n; i++)  
 printf("\nМинимальный элемент %d-ой строки — %d.", i + 1, min\_of\_string(f, i));  
 return 0;  
}

1. 